EXHIBIT B

Recognized as an
American National Standard (ANSI)

IEEE
Std 610.12-1990
**(Revision and redesignation of
IEEE Std 729-1983)**

IEEE Standard Glossary of Software Engineering Terminology

Sponsor
**Standards Coordinating Committee
of the
IEEE Computer Society**

Approved September 28, 1990
IEEE Standards Board

Abstract: IEEE Std 610.12-1990, *IEEE Standard Glossary of Software Engineering Terminology,* identifies terms currently in use in the field of Software Engineering. Standard definitions for those terms are established.
Keywords: Software engineering; glossary; terminology; definitions; dictionary

Print: ISBN 1-55937-067-X, SH13748
PDF: ISBN 0-7381-0391-8, SS13748

Foreword

(This Foreword is not a part of IEEE Std 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology.)

The computer field is continuing to expand. New terms are being generated and new meanings are being adopted for existing terms. The IEEE Computer Dictionary project was undertaken to document this vocabulary. Its purpose is to identify terms currently in use in the computer field and to establish standard definitions for these terms. The dictionary is intended to serve as a useful reference for those in the computer field and for those who come into contact with computers either through their work or in their everyday lives.

The dictionary is being developed as a set of subject-area glossaries covering Computer Architecture, Computer Processors, Computer Storage, Software Engineering, Mathematics of Computing, Theory of Computation, Computer Applications, Artificial Intelligence, Data Management, Image Processing and Pattern Recognition, Modeling and Simulation, Computer Graphics, Computer Networking, Computer Languages, and Computer Security and Privacy. This glossary contains the terms related to Software Engineering. It updates IEEE Std 729-1983, IEEE Standard Glossary of Software Engineering Terminology (ANSI).

Every effort has been made to use definitions from established standards in this dictionary. When existing standards were found to be incomplete, unclear, or inconsistent with other entries in the dictionary, however, new, revised, or composite definitions have been developed.
At the time this glossary was approved, the following people formed the steering committee of the Computer Dictionary working group:

Jane Radatz, Chairperson, *Software Engineering Glossary*

Other subgroup leaders:

Anne Geraci	Louise McMonegal	Paul Wilson
Freny Katki	Bennett Meyer	Mary Yee
Dr. John Lane	Dr. Hugh Porteous	John Young
	Dr. Fredrick Springsteel	

Other working group members who contributed to this glossary were as follows:

Russell J. Abbott	John D. Earls	Dr. José Muñoz
A. Frank Ackerman	Mary Forcht-Tucker	Geraldine Neidhart
Roger R. Baldwin	David Gelperin	Mary Rasmussen
H. Ronald Berlack	Al Gillen	Max Schindler
J. David Bezek	Shirley A. Gloss-Soler	Paul Schmid
James H. Bradley	John A. Goetz	Leonard W. Seagren
Kathleen L.Briggs	David A. Gustafson	Sonja Peterson Shields
Homer C. Carney	Virl Haas	Kevin Smith
Susann Chonoles	James Ingram	Wayne Smith
Taz Daughtrey	Gary S. Lindsay	Paul U. Thompson
Frank J. Douglas	Robert McBeth	Andrew H. Weigel
William P. Dupras	Alicia McCurdy	W. Martin Wong

Special representatives to the Computer Dictionary working group were as follows:

Frank Jay, *Advisor, IEEE Standards Office*

Rollin Mayer, *Liaison, Accredited Standards Committee X3K5*

IEEE Standard Glossary of Software Engineering Terminology

1. Scope

This glossary defines terms in the field of Software Engineering. Topics covered include addressing; assembling, compiling, linking, loading; computer performance evaluation; configuration management; data types; errors, faults, and failures; evaluation techniques; instruction types; language types; libraries; microprogramming; operating systems; quality attributes; software documentation; software and system testing; software architecture; software development process; software development techniques; and software tools.

Every effort has been made to include all terms that meet these criteria. Terms were excluded if they were considered to be parochial to one group or organization; company proprietary or trademarked; multi-word terms whose meaning could be inferred from the definitions of the component words; or terms whose meaning in the computer field could be directly inferred from their standard English meaning.

This glossary is an update and expansion of IEEE Std 729-1983, IEEE Standard Glossary of Software Engineering Terminology (ANSI) [3].[1] It increases the number of terms from approximately 500 to 1300, and updates or refines the definitions of many terms included in the initial glossary. A few terms that were included in the initial glossary have been moved to other glossaries in the 610 series. Some definitions have been recast in a system, rather than software, context. Every effort has been made to preserve the fine work that went into the initial glossary.

2. Glossary Structure

Entries in the glossary are arranged alphabetically. An entry may consist of a single word, such as "software," a phrase, such as "test case," or an acronym, such as "CM." Phrases are given in their natural order (test plan) rather than in reversed order (plan, test).

Blanks precede all other characters in alphabetizing. Hyphens and slashes are treated as blanks. Alternative spellings are shown in parentheses.

If a term has more than one definition, the definitions are numbered. In most cases, noun definitions are given first, followed by verb and adjective definitions as applicable. Examples, notes, and illustrations have been added to clarify selected definitions.

The following cross-references are used to show a term's relationship to other terms in the dictionary:

- *Contrast with* refers to a term with an opposite or substantially different meaning.
- *Syn* refers to a synonymous term.
- *See also* refers to a related term.
- *See* refers to a preferred term or to a term where the desired definition can be found.

The word "deprecated" indicates a term or definition whose use is discouraged because such use is obsolete, misleading, or ambiguous. "DoD" refers to usage by the U.S. Department of Defense.

3. Definitions for Software Engineering Terms

1GL. Acronym for **first generation language.** *See:* **machine language.**

2GL. Acronym for **second generation language.** *See:* **assembly language.**

3GL. Acronym for **third generation language.** *See:* **high order language.**

[1]Numbers in brackets correspond to those in the Bibliography in Section 4.

developmental configuration. In configuration management, the software and associated technical documentation that define the evolving configuration of a computer software configuration item during development. *Note:* The developmental configuration is under the developer's control, and therefore is not called a baseline. *Contrast with:* **allocated baseline; functional baseline; product baseline.**

deviation. (1) A departure from a specified requirement.
(2) A written authorization, granted prior to the manufacture of an item, to depart from a particular performance or design requirement for a specific number of units or a specific period of time. *Note:* Unlike an engineering change, a deviation does not require revision of the documentation defining the affected item. *See also:* **configuration control.** *Contrast with:* **engineering change; waiver.**

device. A mechanism or piece of equipment designed to serve a purpose or perform a function.

DFD. Acronym for **data flow diagram.**

diagnostic. Pertaining to the detection and isolation of faults or failures; for example, a diagnostic message, a diagnostic manual.

diagnostic manual. A document that presents the information necessary to execute diagnostic procedures for a system or component, identify malfunctions, and remedy those malfunctions. Typically described are the diagnostic features of the system or component and the diagnostic tools available for its support. *See also:* **installation manual; operator manual; programmer manual; support manual; user manual.**

diagonal microinstruction. A microinstruction capable of specifying a limited number of simultaneous operations needed to carry out a machine language instruction. *Note:* Diagonal microinstructions fall, in size and functionality, between horizontal microinstructions and vertical microinstructions. The designation "diagonal" refers to this compromise rather than to any physical characteristic of the microinstruction. *Contrast with:* **horizontal microinstruction; vertical microinstruction.**

differential dump. *See:* **change dump.**

digraph. *See:* **directed graph.**

direct address. An address that identifies the storage location of an operand. *Syn:* **one-level address.** *Contrast with:* **immediate data; indirect address; *n*-level address.** *See also:* **direct instruction.**

direct insert subroutine. *See:* **open subroutine.**

direct instruction. A computer instruction that contains the direct addresses of its operands. *Contrast with:* **immediate instruction; indirect instruction.** *See also:* **absolute instruction; effective instruction.**

directed graph. A graph (sense 2) in which direction is implied in the internode connections. *Syn:* **digraph.** *Contrast with:* **undirected graph.**

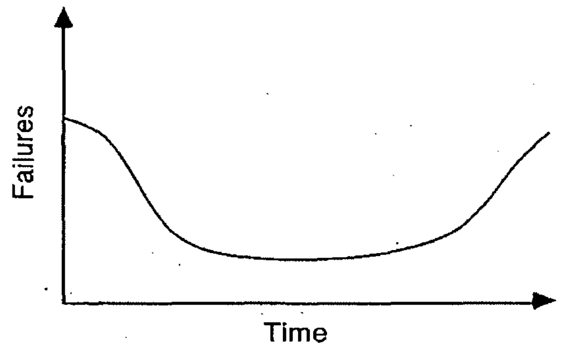

Fig 11
Graph (1)

(2) A diagram or other representation consisting of a finite set of nodes and internode connections called edges or arcs. *See also:* **block diagram; box diagram; bubble chart; directed graph; flowchart; input-process-output chart; structure chart.**

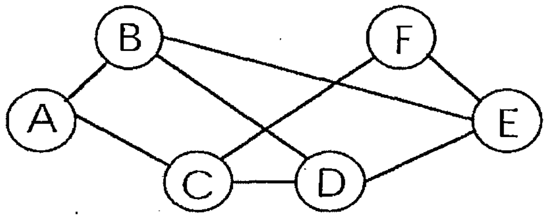

Fig 12
Graph (2)

Grosch's law. A guideline formulated by H. R. J. Grosch, stating that the computing power of a computer increases proportionally to the square of the cost of the computer. *See also:* **computer performance evaluation.**

halt. (1) Most commonly, a synonym for **stop.**
(2) Less commonly, a synonym for **pause.**

hard failure. A failure that results in complete shutdown of a system. *Contrast with:* **soft failure.**

hardware. Physical equipment used to process, store, or transmit computer programs or data. *Contrast with:* **software.**

hardware configuration item (HWCI). An aggregation of hardware that is designated for configuration management and treated as a single entity in the configuration management process. *Contrast with:* **computer software configuration item.** *See also:* **configuration item.**

hardware design language (HDL). A language with special constructs and, sometimes, verification protocols, used to develop, analyze, and document a hardware design. *See also:* **program design language.**